

DEBUGGING SYSTEM

Publication number: JP7160530 (A)

Publication date: 1995-06-23

Inventor(s): FUKUDA MASAHIRO +

Applicant(s): NEC CORP +

Classification:

- international: G06F11/28; G06F9/46; G06F9/48; G06F11/28; G06F9/46; (IPC1-7): G06F11/28; G06F11/28; G06F9/46

- European:

Application number: JP19930304356 19931203

Priority number(s): JP19930304356 19931203

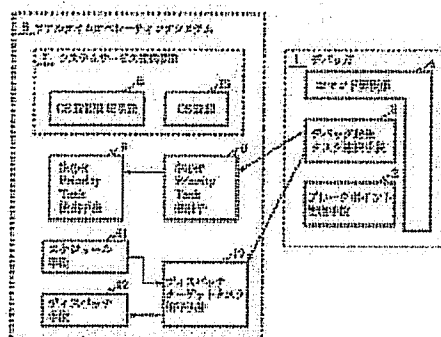
Also published as:

JP2643804 (B2)

Abstract of JP 7160530 (A)

PURPOSE:To debug a task in a ready state and a break state and to keep the state of a ready queue and OS resources at that time point even after debugging is finished.

CONSTITUTION:Corresponding to the command of a user for debugging the target task in the ready state inputted in the break state, a debugger 1 sets a break point to the execution restart point of this target task, applies a dummy system call to a real-time OS 1, sets a Super Priority Task identifier and sets this target task to a dispatch target task holding means 10. When the Super Priority Task identifier is detected by a Super Priority Task detecting means 9, a dispatch means 12 dispatches the set target task.



Data supplied from the *espacenet* database — Worldwide

(19)日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開平7-160530

(43)公開日 平成7年(1995)6月23日

(51)Int.Cl. ⁶	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 11/28	A	9290-5B		
	3 1 5 A	9290-5B		
9/46	3 3 0 C	7629-5B		

審査請求 有 請求項の数5 O L (全 9 頁)

(21)出願番号 特願平5-304356

(22)出願日 平成5年(1993)12月3日

(71)出願人 000004237

日本電気株式会社
東京都港区芝五丁目7番1号

(72)発明者 福田 政広

東京都港区芝五丁目7番1号 日本電気株
式会社内

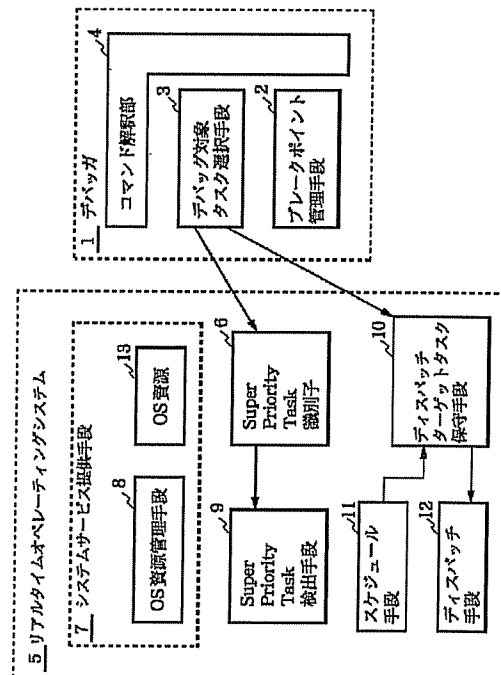
(74)代理人 弁理士 井出 直孝

(54)【発明の名称】 デバッグ方式

(57)【要約】

【目的】 レディ状態およびブレイク状態のタスクをデバッグでき、かつデバッグ終了後もレディキューの状態およびその時点のOS資源を維持できる。

【構成】 デバッガ1は、ブレイク状態のとき入力したレディ状態のターゲットタスクをデバッグするユーザのコマンドにより、このターゲットタスクの実行再開点にブレイクポイントをセットし、リアルタイムOS1にダミーシステムコールを与え、SuperPriorityTask 識別子をセットし、ディスパッチターゲットタスク保持手段10にこのターゲットタスクをセットする。SuperPriorityTask 検出手段9がSuperPriorityTask 識別子を検出したときにディスパッチ手段12はセットされたターゲットタスクをディスパッチする。



【特許請求の範囲】

【請求項1】 システムサービスを提供する提供手段、タスクを選出する順序を定めるスケジュール手段、ディスパッチするターゲットタスクを保持するディスパッチターゲットタスク保持手段およびこのディスパッチターゲットタスク保持手段からのターゲットタスクをディスパッチするディスパッチ手段を含むリアルタイムオペレーティングシステムと、デバッグとを備えたデバッグ方式において、

前記デバッグから前記リアルタイムオペレーティングシステムに対して、運用上のプライオリティよりさらに優先度を高くしてデバッグを行うためのスーパープライオリティ識別子をセットする手段を設けたことを特徴とするデバッグ方式。

【請求項2】 前記リアルタイムオペレーティングシステムに、前記スーパープライオリティ識別子がセットされているか否かを検出し検出結果を出力する検出手段を含み、

前記デバッグは、タスクがブレークポイントで停止状態のときレディ状態にあるターゲットタスクをデバッグするユーザのコマンドを入力してデバッグ対象のターゲットタスクを指定しこの指定したターゲットタスクをディスパッチするダミーシステムコールを前記システムサービス提供手段に出力するコマンド解釈部と、この指定されたターゲットタスクの実行再開点にブレークポイントをセットするブレークポイント管理手段と、この指定されたターゲットタスクを選択して前記ディスパッチターゲットタスク保持手段にセットし前記スーパープライオリティ識別子をセットする選択手段とを含む請求項1記載のデバッグ方式。

【請求項3】 前記システムサービス提供手段は前記ダミーシステムコールに基づき処理を行う手段を含み、前記ディスパッチ手段は前記検出結果がセットされている場合に前記スケジュール手段をバイパスしてディスパッチターゲットタスク保持手段にセットされたターゲットタスクをディスパッチする手段を含む請求項2記載のデバッグ方式。

【請求項4】 請求項1記載のデバッグ方式において、前記リアルタイムオペレーティングシステムに、前記スーパープライオリティ識別子がセットされているか否かを検出し検出結果を出力する検出手段を含み、前記デバッグは、タスクがブレークポイントで停止状態のときレディ状態にないターゲットタスクをデバッグするユーザのコマンドを入力してデバッグ対象のターゲットタスクを指定し前記システムサービス提供手段にタスクの実行を再開させるコマンド解釈部と、この指定されたターゲットタスクに実行再開点のブレークポイントをセットするブレークポイント管理手段と、この指定されたターゲットタスクを選択して前記スーパープライオリティタスク保持手段にセットする選択手段とを含むことを

特徴とするデバッグ方式。

【請求項5】 前記システムサービス提供手段は、タスクをレディキューに接続するオペレーティング資源管理手段を含み、

前記オペレーティングサービス資源管理手段は、前記システムサービス提供手段がシステムサービスを行った各時点で前記スーパープライオリティタスク保持手段の内容と前記レディキューに接続されたタスクとを比較し比較結果が一致した場合に前記スーパープライオリティ識別子をセットする手段を含み、

前記検出手段は前記検出結果がセットされている場合には前記スーパープライオリティタスク保持手段の内容を前記ディスパッチターゲットタスク保持手段にセットする手段を含み、

前記スケジュール手段は前記検出結果がセットされていない場合には次にディスパッチするタスクを前記ディスパッチターゲットタスク保持手段にセットする手段を含み、

前記ディスパッチ手段は前記ディスパッチターゲットタスク保持手段にセットされたタスクをディスパッチする手段を含む請求項4記載のデバッグ方式。

【発明の詳細な説明】

【0001】

【産業上の利用分野】 本発明は、リアルタイムオペレーティングが稼働する組み込みシステムをデバッグするデバッグ方式に利用する。特に、リアルタイムオペレーティングシステム(RTOS)に状態遷移が発生した時点でブレーク状態およびレディ状態の任意のタスクをデバッグするデバッグ方式に関するものである。

【0002】

【従来の技術】 図5はデバッグ方式のリアルタイムオペレーティングシステムのタスクの状態遷移図である。図6はデバッグ方式のリアルタイムオペレーティングシステムのOS資源の管理状態を示す図である。図7は従来例のデバッグ方式のリアルタイムオペレーティングシステムのレディキューのキュー変化を示す図である。図5において、システムコールの略語のフルスペリングは次のとおりである。cre_tsk() はcreate task、sta_tsk() はstart task、del_tsk() はdelete task、ext_tsk() はexit task、exd_tsk() はexit and delete task、ter_tsk() はterminate task、sus_tsk() はsuspend task、rsm_tsk() はresume task および wup_tsk() はwake up taskを示す。また、図6において、tcb_adr() はget task-control-block addressを示す。

【0003】 従来、デバッグ方式は、特に組み込みシステムなどで使用される目的で開発されたリアルタイムオペレーティングシステム(以下、リアルタイムOSと云う。)の場合に、デッドラインを持つ複数のアプリケーションタスクを並行動作させながら、目的の処理を行

う。

【0004】複数のタスクを並行動作させるOSには様々な種類がある。OSによってタスクが取得する状態の数や種類が異なる。その中の一つの例としてここでは、ITRON OS (アイトロンOS、industrial TRON OS) で採用されている状態を説明する。ITRON OSでは、タスクは以下の七つの状態の内のどれかになっている。

【0005】実効状態 RUN

実効可能状態 READY

待ち状態 WAIT

二重待ち状態 WAIT-SUSPEND

強制待ち状態 SUSPEND

休止状態 DORMANT

未登録状態 NON-EXISTENCE

前記の状態がどのように変化するかを図5に示す。

【0006】組み込みシステムにOSを使用して移動させる場合に、OSが管理するさまざまな資源(タスク、フラグ、セマフォ、メールボックス、メモリアル、メモリブロック)等は、OSによって様々な状態の間を遷移させられる。かつ、必ずOSによって一貫性が保たれながら状態遷移する。

【0007】例として、図6にITRON準拠のOSであるRX116においてOS資源がどのように管理されているかの概念図を示す。タスクは、RUN、READY、SLEEP (WAITに含まれる)、SUSPEND、WAIT-SUSPEND等の内のどれかの状態になっている。前記の状態遷移途中の状態で存在することはあり得ない。

【0008】中央処理装置(以下、CPUと云う。)が実行しているタスクは必ず、RUN状態である。よって、再実行、ステップ実行等のデバッグ操作を行うために、タスクは必ずRUN状態になっていなければならない。

【0009】SUSPEND状態やWAIT状態のタスクを強制的に実行することは危険である。OSが自分の資源を操作しているときに、SUSPEND状態やWAIT状態のタスクが実行されることは想定していない。OSは自分が管理しているOS資源が他の要因によりOS管理状態以外の状況に変更されている可能性は考慮していない。無駄なチェックを省き、性能を向上させるためである。

【0010】OS管理する状態以外にならないようにするために、デバッグは前記のようなタスクを実行させない。

【0011】このような条件の元では、ターゲットタスクが最高プライオリティになるのを待たねばデバッグができない。しかし、このようなデバッグ方法が使えないのは、他のタスクに不具合がない場合のみである。他のタスクに不具合がある場合には、ターゲットタスクが最高

プライオリティになるより先に組み込みシステムが暴走し、結果としてターゲットタスクのデバッグが行えなくなる。

【0012】そこで、ターゲットタスクを最高プライオリティにするために、従来はプライオリティを高くする必要があった。プライオリティを高くすることによって、RUN状態に移行させデバッグ停止状態にする。

【0013】ところが、一度プライオリティを高くすると、元に戻すときに、元あったプライオリティキューの再後尾に接続されるので、最初の状態から変化する欠点があった。

【0014】このことを詳細に説明する。例えば、図7(A)の場合にTaskCをデバッグすることを考える。ここで、TaskCをデバッグするためにchg - pri (change priority) システムコールを発行してプライオリティを上げると図7(B)のようになる。OSによって、TaskCが選択され、デバッグのための実行が可能となる。デバッグが終了した後に、もとのプライオリティに戻すと図7(C)のようになる。

【0015】このようにしてTaskCがプライオリティ3のキューの最後尾になる。TaskCを前記キューの最初に移動するためには、rot - rdq (rotate ready queue) システムコールを六回発行するか、TaskDからTaskIに関してTaskCに行ったのと同じ操作を繰り返さなければならない。その場合のchg - pri システムコール発行回数は12回である。システムコール発行する回数を決するためには、あらかじめレディキューに接続されているタスクの数を調べなければならない。

【0016】ここでは、キューの先頭にあるタスクを例として挙げたので、元の状態に戻す手段が存在した。しかし、図7(D)の位置にあるタスクをデバッグした場合に元の状態に戻すことはさらに困難になる。

【0017】

【発明が解決しようとする課題】このような従来例のデバッグ方式では、デバッグ途上の組み込みシステムをデバッグする場合に、ターゲットタスクが最高プライオリティになるのを待っていると、ターゲットタスク以外のタスクが不具合を持っている場合に組み込みシステムが暴走して、結果としてターゲットタスクのデバッグが不可能になる欠点があった。特に、リアルタイムアプリケーションにおいては、タスク間の起動順序等によって動作が大きく変化するために、いちがいに不具合をもってタスクを停止すれば済むわけではない。

【0018】また、前述のように、従来のシステムコールのみを組み合わせた場合に、レディキューの順序を元の位置に回復するための手段が存在しなかったり、可能であったとしても多くのオーバーヘッドが生じる。

【0019】本発明は前記の欠点を解決するもので、レディ状態のタスクおよび状態遷移が発生した時点でのブレーク状態のタスクをデバッグすることができ、かつデ

バッグ終了後もレディキューの順序が変化せずまたレディ状態になった時点のオペレーションシステム資源を維持しながらデバッグできるデバッグ方式を提供することを目的とする。

【0020】

【課題を解決するための手段】本発明の第一の観点は、システムサービスを提供する提供手段、タスクを選出する順序を定めるスケジュール手段、ディスパッチするターゲットタスクを保持するディスパッチターゲットタスク保持手段およびこのディスパッチターゲットタスク保持手段からのターゲットタスクをディスパッチするディスパッチ手段を含むリアルタイムオペレーティングシステムと、デバッグとを備えたデバッグ方式において、前記デバッグから前記リアルタイムオペレーティングシステムに対して、運用上のプライオリティよりさらに優先度を高くしてデバッグを行うためのスーパープライオリティ識別子を設定する手段を設けたことを特徴とする。

【0021】また、本発明は、前記リアルタイムオペレーティングシステムに、前記スーパープライオリティ識別子がセットされているか否かを検出し検出結果を出力する検出手段を含み、前記デバッグは、タスクがブレイクポイントで停止状態のときレディ状態にあるターゲットタスクをデバッグするユーザのコマンドを入力してデバッグ対象のターゲットタスクを指定しこの指定したターゲットタスクをディスパッチするダミーシステムコールを前記システムサービス提供手段に出力するコマンド解釈部と、この指定されたターゲットタスクの実行再開点にブレイクポイントをセットするブレイクポイント管理手段と、この指定されたターゲットタスクを選択して前記ディスパッチターゲットタスク保持手段にセットし前記スーパープライオリティ識別子をセットする選択手段とを含むことができる。

【0022】さらに、本発明は、前記システムサービス提供手段は前記ダミーシステムコールに基づき処理を行う手段を含み、前記ディスパッチ手段は前記検出結果がセットされている場合に前記スケジュール手段をバイパスしてディスパッチターゲットタスク保持手段にセットされたターゲットタスクをディスパッチする手段を含むことができる。

【0023】本発明の第二の観点は、前記デバッグ方式において、前記リアルタイムオペレーティングシステムに、前記スーパープライオリティ識別子がセットされているか否かを検出し検出結果を出力する検出手段を含み、前記デバッグは、タスクがブレイクポイントで停止状態のときレディ状態にないターゲットタスクをデバッグするユーザのコマンドを入力してデバッグ対象のターゲットタスクを指定し前記システムサービス提供手段にタスクの実行を再開させるコマンド解釈部と、この指定されたターゲットタスクに実行再開点のブレイクポイントをセットするブレイクポイント管理手段と、この指定され

たターゲットタスクを選択して前記スーパープライオリティタスク保持手段にセットする選択手段とを含むことを特徴とする。

【0024】また、本発明は、前記システムサービス提供手段は、タスクをレディキューに接続するオペレーティング資源管理手段を含み、前記オペレーティングサービス資源管理手段は、前記システムサービス提供手段がシステムサービスを行った各時点で前記スーパープライオリティタスク保持手段の内容と前記レディキューに接続されたタスクとを比較し比較結果が一致した場合に前記スーパープライオリティ識別子をセットする手段を含み、前記検出手段は前記検出結果がセットされている場合には前記スーパープライオリティタスク保持手段の内容を前記ディスパッチターゲットタスク保持手段にセットする手段を含み、前記スケジュール手段は前記検出結果がセットされていない場合には次にディスパッチするタスクを前記ディスパッチターゲットタスク保持手段にセットする手段を含み、前記ディスパッチ手段は前記ディスパッチターゲットタスク保持手段にセットされたタスクをディスパッチする手段を含むことができる。

【0025】

【作用】デバッグは、タスクがブレイクポイントで停止しているときにレディ状態のターゲットタスクをデバッグするユーザのコマンドを入力したとき、リアルタイムオペレーティングシステムにスーパープライオリティ識別子をセットし、また、ディスパッチターゲットタスク保持手段にデバッグ対象ターゲットをセットし、さらにダミーシステムコールを提供手段に与える。リアルタイムオペレーティングシステムでは、検出手段でスーパープライオリティ識別子がセットされているか否かを検出し、検出した場合にはディスパッチターゲットタスク保持手段に保持されたターゲットタスクをディスパッチし、ターゲットタスクの実行開始点でブレイクが発生しターゲットタスクのデバッグが開始する。

【0026】また、デバッグは、タスクがブレイクポイントで停止しているときにレディ状態でないターゲットタスクをデバッグするユーザのコマンドを入力したとき、スーパープライオリティタスク保持手段にターゲットタスクをセットしてターゲットシステムの実行を再開させる。リアルタイムオペレーティングシステムでは、オペレーティングシステム資源管理手段でシステムサービスを行った各時点でスーパープライオリティタスク保持手段の内容とレディキューの内容とを比較し一致した場合にはスーパープライオリティ識別子をセットする。検出手段は、スーパープライオリティ識別子がセットされているか否かを検出し、検出した場合にはスーパープライオリティタスク保持手段の内容をディスパッチターゲットタスク保持手段にセットする。検出しない場合にはスケジュール手段は次にディスパッチするタスクをディスパッチターゲット保持手段にセットする。ディスパッチ手段は

ディスパッチターゲットタスク保持手段の内容をディスパッチし、ディスパッチされたタスクがスーパープライオリティタスクでないときにはシステムサービスを行ない、スーパープライオリティタスクのときにはターゲットタスクの実行開始点でブレークが発生しターゲットタスクのデバッグを開始する。

【0027】以上によりレディ状態のタスクおよび状態遷移が発生した時点でのブレーク状態のタスクをデバッグすることができ、かつデバッグ終了後もレディキューの順序が変化せずまたレディ状態になった時点のオペレーションシステム資源を維持しながらデバッグできる。

【0028】

【実施例】本発明の実施例について図面を参照して説明する。

【0029】図1は本発明一実施例デバッグ方式のブロック構成図である。図1において、デバッグ方式は、システムサービスを提供する提供手段7、タスクを選出する順序を定めるスケジュール手段11、ディスパッチするターゲットタスクを保持するディスパッチターゲットタスク保持手段10およびディスパッチターゲットタスク保持手段10からのターゲットタスクをディスパッチするディスパッチ手段12を含むリアルタイムオペレーティングシステム5と、デバッグ1とを備える。

【0030】ここで本発明の特徴とするところは、デバッグ1からリアルタイムオペレーティングシステム5に対して、運用上のプライオリティよりさらに優先度を高くしてデバッグを行うためのスーパープライオリティ識別子(SuperPriorityTask 識別子)6をセットする手段を設けたことにある。

【0031】また、リアルタイムオペレーティングシステム5に、スーパープライオリティ識別子がセットされているか否かを検出し検出結果を出力する検出手段としてSuperPriorityTask 検出手段9を含み、デバッグ1は、タスクがブレークポイントで停止状態のときレディ状態にあるターゲットタスクをデバッグするユーザのコマンドを入力してデバッグ対象のターゲットタスクを指定しこの指定したターゲットタスクをディスパッチするダミーシステムコールをシステムサービス提供手段7に出力するコマンド解釈部4と、この指定されたターゲットタスクの実行再開点にブレークポイントをセットするブレークポイント管理手段2と、この指定されたターゲットタスクを選択してディスパッチターゲットタスク保持手段10にセットしSuperPriorityTask 識別子6をセットする選択手段としてデバッグ対象タスク選択手段3とを含む。

【0032】さらに、システムサービス提供手段7は前記ダミーシステムコールに基づき処理を行う手段を含み、ディスパッチ手段12は前記検出結果がセットされている場合にスケジュール手段11をバイパスしてディスパッチターゲットタスク保持手段10にセットされた

ターゲットタスクをディスパッチする手段を含む。

【0033】このような構成のデバッグ方式の動作について説明する。

【0034】図2は本発明のデバッグ方式の動作を示す図である。図2において、任意のターゲットタスクを指定し、指定したターゲットタスクがデバッグ停止状態になるまでのフローを示す。まず、どれかのタスクがブレークポイントで停止している状態が図2(1)である。

【0035】この時点でユーザは、コマンドによってレディ状態にあるターゲットタスクを指定することができるようになる。それが、図2(2)である。ユーザがコマンドでターゲットタスクを指定すると、デバッグ1はターゲットタスクの実行再開点にブレークポイントをセットする。デバッグを停止させるためである。

【0036】その後、デバッグ1がSuperPriorityTask 識別子6をセットし、ディスパッチターゲットタスク保持手段10にターゲットタスクをセットする〔図2(3)〕。この後に、ターゲットタスクをディスパッチさせるためデバッグ1がダミーシステムコールを発行する〔図2(4)〕。

【0037】ダミーシステムコールを発行すると、指定されたシステムコールがシステムサービス提供手段7にて処理され、SuperPriorityTask 検出手段9はSuperPriorityTask 識別子6を検出する。SuperPriorityTask 識別子6が検出されると、スケジュール手段11をバイパスし、ディスパッチ手段12が実行される〔図2(5)〕。

【0038】ディスパッチ手段12はディスパッチターゲットタスク保持手段10に保持されたターゲットタスクをディスパッチする〔図2(6)〕。

【0039】ディスパッチが発生し、ターゲットタスクが実行を開始する。実行を開始した時点で、図2(2)にてセットしたブレークポイントに達して停止する〔図2(7)〕。

【0040】この状態がデバッグ停止状態である。デバッグは安全に、すなわちOSの管理状態の元でタスクをデバッグ可能となる〔図2(8)〕。

【0041】図3は本発明他の実施例デバッグ方式のブロック構成図である。図4は本発明他の実施例デバッグ方式の動作を示す図である。図3および図4を参照して、指定したタスク状態が遷移したときに、その時点でデバッグできる例について説明する。

【0042】まず、図3において、1AはRTOS用のデバッグ、2はブレークポイント管理手段、3Aはデバッグ対象タスク選択手段、4はコマンド解釈部、5Aはリアルタイムオペレーティングシステム、6はSuperPriorityTask 識別子、14はSuperPriorityTask 保持手段、7Aはシステムサービス提供手段(タスク管理、フラグ管理、セマフォ管理、メールボックス管理、メモリプール管理)、8AはOS資源管理手段(レディキュー

管理、タイマキュー管理、OS資源待ち管理)、9AはSuperPriorityTask 検出手段、10はディスパッチターゲットタスク保持手段、11はスケジュール手段および12はディスパッチ手段、13はOS資源である。OS資源13には、図6に示すようにタスク、フラグ、セマフォ、メールボックス、メモリアル等が含まれる。

【0043】図4に、レディ状態でない任意のターゲットタスクがレディ状態に移移する時点が検出され、デバッグ停止状態になるまでのフローを示す。図4において、まず、どれかのタスクがブレークポイントで停止している状態が図4(1)である。どれかのタスクがブレークポイントで停止した時点から、ユーザのデバッグ操作が開始される〔図4(1)〕。レディ状態にないタスク中で、レディ状態になったことを検出したいデバッグ対象のターゲットタスクを、ユーザが入力するコマンドに従って、デバッグ1がSuperPriorityTask 保持手段14にセットする。さらに、ユーザが入力するコマンドに従って、デバッグ1がターゲットタスクの実行開始点にブレークポイントをセットする〔図4(2)〕。

【0044】ユーザが入力するコマンドに従い、ターゲットシステムの実行を再開する。この時点からターゲットシステム側で並行プログラミングされたタスクが動作を開始する〔図4(3)〕。

【0045】並行プログラミングされたタスクが動作を開始すると、システムサービスを行った各時点で、レディキュー管理手段を含むOS資源管理手段8Aが、システムサービスを行った結果としてOS資源管理手段8Aがレディキューに接続したタスクとSuperPriorityTask 保持手段14の内容とを比較し、それが同一の場合にはSuperPriorityTask 識別子6をセットする〔図4(4)〕。

【0046】SuperPriorityTask 検出手段9Aによって、SuperPriorityTask 識別子6がセットされているかどうかをチェックする。セットされている場合には〔図4(5)〕、SuperPriorityTask 保持手段14の内容をディスパッチターゲットタスク保持手段10にコピーする〔図4(6)〕。コピー後にスケジュール手段11をスキップし、ディスパッチ手段12へ移行する。セットされていない場合には〔図4(5)〕、スケジュール手段11へ移行する。スケジュール手段11では、レディキュー状態のタスクの中の最高プライオリティキュー(SuperPriorityTask とは異なる)に接続されたタスクの中で最も最初にプライオリティキューに接続されたタスクをディスパッチターゲットタスク保持手段10にセットする〔図4(7)〕。

【0047】ディスパッチ手段12は、ディスパッチターゲットタスク保持手段10の内容をディスパッチする〔図4(8)〕。

【0048】ディスパッチ手段12でディスパッチされたタスクがSuperPriorityTask か否かをチェックし、Su

perPriorityTask の場合には〔図4(9)〕、ターゲットタスクの実行開始点でブレーク発生し〔図4(10)〕、ターゲットタスクのデバッグを開始する〔図4(11)〕。

【0049】ディスパッチ手段12でディスパッチされたタスクがSuperPriorityTask でない場合には〔図4(9)〕、図4(4)に戻り動作を繰り返す。

【0050】ここで、リアルタイムオペレーティングシステム用のデバッグはすべてターゲットシステム内に存在する必要がある。例えば、リアルタイムオペレーティングシステム用のデバッグをホスト部とターゲット部とに分割しコマンド解釈部はホスト側に移しても良い。

【0051】

【発明の効果】以上説明したように、本発明は、レディ状態のタスクであれば、レディキューの中のいかなる場所においてもレディキューの状態を変化させずにデバッグを実行することができる。また、タスクが待ち状態からレディ状態になった時点でシステムを停止することができ、ターゲットタスクがレディ状態になった時点のシステムの状態(OS資源の状態)を維持しながらのデバッグができる。さらに、該当タスクを実行状態にし実行させてから中断するために、ブレークさせたタスクはカーネル内部状態と一致したままデバッグができる優れた効果がある。

【図面の簡単な説明】

【図1】本発明一実施例デバッグ方式のブロック構成図。

【図2】本発明のデバッグ方式の動作を示す図。

【図3】本発明他の実施例デバッグ方式のブロック構成図。

【図4】本発明他の実施例デバッグ方式の動作を示す図。

【図5】デバッグ方式のリアルタイムオペレーティングシステムのタスクの状態遷移図。

【図6】デバッグ方式のリアルタイムオペレーティングシステムのOS資源の管理状態を示す図。

【図7】従来例のデバッグ方式のリアルタイムオペレーティングシステムのレディキューのキュー変化を示す図。

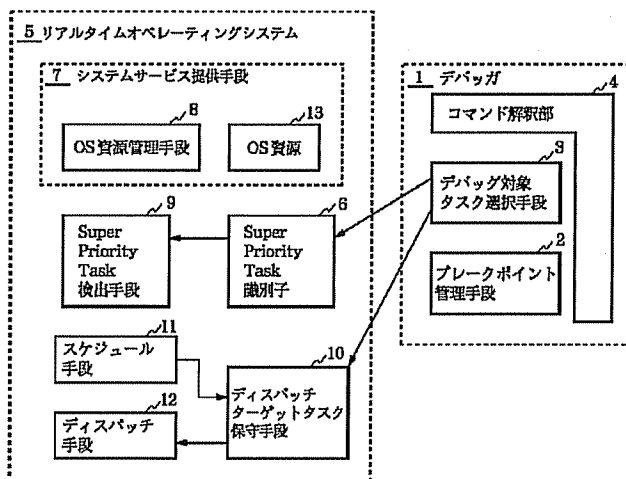
【符号の説明】

- 1、1A デバッグ(RTOS用)
- 2 ブレークポイント管理手段
- 3、3A デバッグ対象タスク選択手段
- 4、4A コマンド解釈部
- 5、5A リアルタイムオペレーティングシステム(RTOS)
- 6 SuperPriorityTask 識別子
- 7、7A システムサービス提供手段
- 8、8A OS資源管理手段
- 9、9A SuperPriorityTask 検出手段

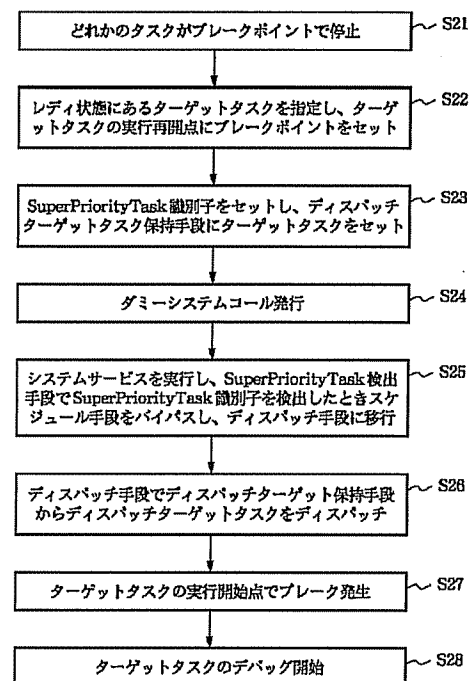
10 ディスパッチターゲットタスク保持手段
 11 スケジュール手段
 12 ディスパッチ手段
 13 OS資源
 14 SuperPriorityTask 保持手段
 101 RUN状態のタスク
 102 プライオリティが1に接続されたタスク (READY状態)
 103 プライオリティが2に接続されたタスク (READY状態)
 104、105 プライオリティが3に接続されたタスク (READY状態)
 106 フラグ待ちキューに接続されたタスク (WAIT状態)
 107、114、116 セマフォ待ちキューに接続されたタスク (WAIT状態)
 108、113、117 メッセージ待ちキューに接続されたタスク (WAIT状態)
 109 メモリブロック待ちキューに接続されたタスク (WAIT状態)
 110 メッセージ待ちキューに接続されたタスク (WAIT状態)
 111 メモリブロック待ちキューに接続されたタスク (WAIT状態)
 112 登録キューのみに接続されたタスク (DORM

ANT状態)
 115、120、121 タイマキューに接続されたタスク (WAIT状態)
 118 登録キューのみに接続されたタスク (SUSPEND状態)
 119 セマフォ待ちキューに接続されたタスク (WAIT-SUSPEND状態)
 122 スリープしているタスク (WAIT状態)
 123 タイマキューに接続されたタスク (WAIT-SUSPEND状態)
 124 スリープしているタスク (WAIT-SUSPEND状態)
 201、202 タスク122を周期的に起床するためのタイマ管理ブロック
 301 メッセージ
 1000 Task Reg. Queue Header
 1001 Flag Reg. Queue Header
 1002 Semaphore Reg. Queue. Queue Header
 1003 Mail Box Reg. Queue Header
 1004 Mail Pool Reg. Queue Header
 1005 Timer Queue
 2001、2002 Flag
 2003、2004 Semaphore
 2005、2006 Mail Box
 2007、2008 Memory Pool

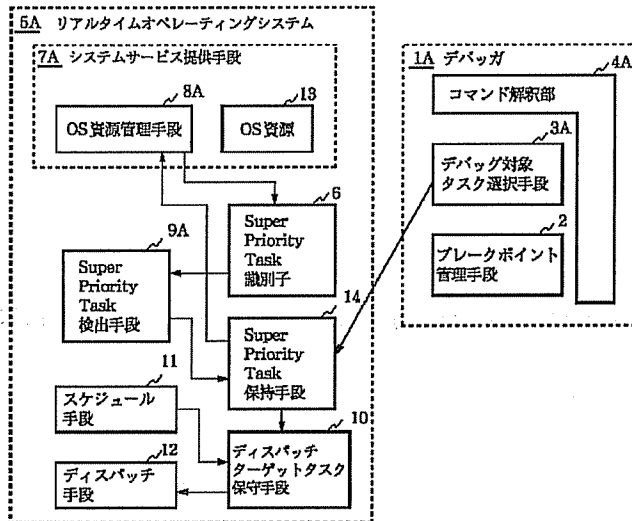
【図1】



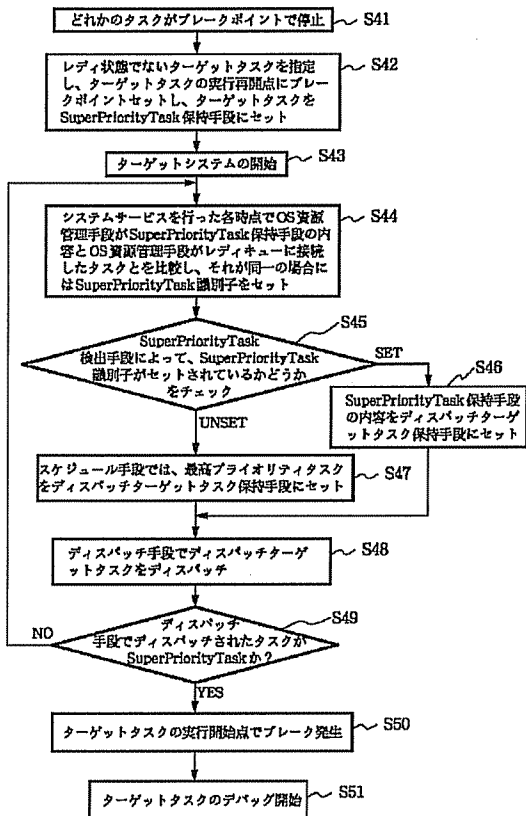
【図2】



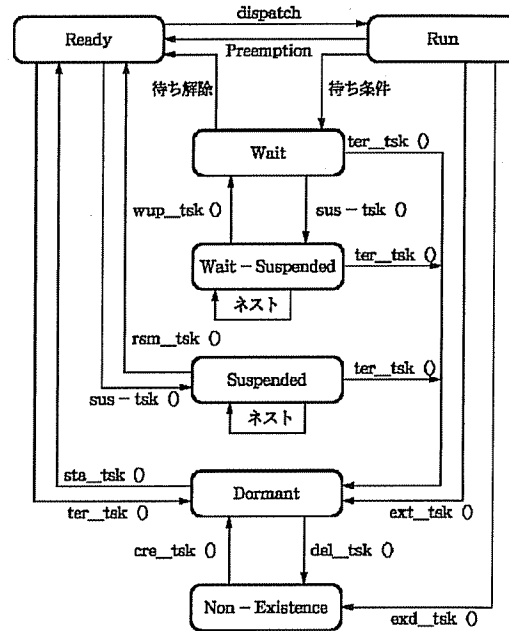
【図3】



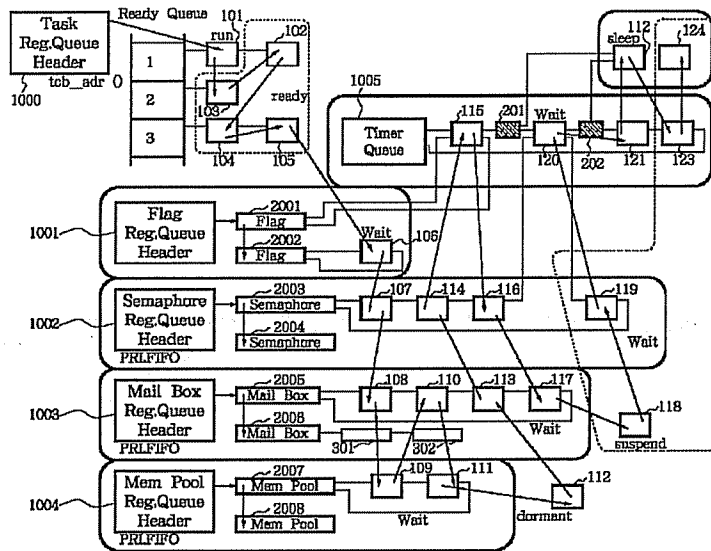
【図4】



【図5】



【図6】



【図7】

